PERFORMANCE EVALUATION OF IPV6 PACKET CLASSFICATION WITH CACHING

Kai-Yuan Ho and Yaw-Chung Chen
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
ycchen@cs.nctu.edu.tw

Abstract

Packet classification can be applied in network security, QoS, routing, network load balancing, bandwidth sharing etc. Algorithms of packet classification are categorized into either hardware-based or software-based solutions. Nowadays packet classification implementations are inefficient in IPv6 network environment because much longer address fields have to be processed. In this paper, we propose schemes that use cache memory to improve the performance of IPv6 packet classification. We evaluate the performance of our schemes through simulation under different cache sizes, architectures, and replacement policies. We use real world IPv6 traffic flows for the experiment, and the numerical results show that our schemes achieve higher than 90% hit rate when cache size is no less than 1024 entries in 4-way associative cache memory architecture, this significantly improves the performance of IPv6 packet classification.

1. Introduction

Packet classification can be applied in network security, QoS, routing, network load balancing, bandwidth sharing etc. Performance of packet classification affects network throughput obviously. With more and more deployments of real-time applications such as VoIP, online games and IPTV, it is essential to provide high speed and low latency packet classification functionality at affordable cost. The problem of packet classification is a generalization of the one-dimensional IP route lookup. Packet classification speeds are mainly limited by memory access latencies, which are about 20 to 50 ns for DRAM, 5 to 20 ns for SRAM, and 1 to 2 ns in on-chip SRAM. Even using on-chip SRAM, the classification process can only accomplish approximately 4 memory lookups for 40Gbps transmission rate. To accommodate today's high-speed network, we propose an approach of caching the recently matched flows, and the process of packet classification must be executed only on a cache miss. In this paper, we evaluate packet classification performance through caching with real IPv6 traffic data, and we compress IPv6 flow ID in order to save cache space. We identify a flow ID using a 5-tuple: <Source Address, Destination Address, Source Port, Destination Port, Protocol>, which contains 296 bits in IPv6 instead of 104 bits in IPv4. Finally, we evaluate miss rate and misclassification rate for our hash functions. In [1][3][4][6], various packet classification algorithms are proposed, and in [2], Gupta et al. survey various classification algorithms. Recent study [7] shows that that the packet arrival process is bursty, rather than Poisson. It implies a very high probability of next packet arrival with the same flow ID. Studies of packet classifications based on flow ID show similar improvement as route lookup by using a cache [5]. Our studies differ from the above in dealing with IPv6 regarding cache performance and hash functions for representation of flow ID.

2. Proposed approach

Random bit-selection is a hash function to generate variable-length bit string. It can be easily implemented in hardware with constant time complexity.

2.1 Samples of traffic data

We use the traffic data provided by MAWI Working Group, whose Traffic Archive [8] has carried out network traffic measurement, analysis, evaluation, and verification from the beginning of the WIDE Project. The traffic samples were captured through *tcpdump* in binary format. We use the sample of the date 2004/03/13 from WIDE-Bone6 as sample 1, which has 2 million packets, 3,507 distinct addresses, and 18,811 distinct flows. We also use the sample of the 2004/05/18 from 6Bone as sample 2. It has 2 million packets, 2,518 distinct addresses, and 17,016 distinct flows. The default buffer size setting in *tcpdump* was too small to recognize all IPv6 extension headers. Some packets with IPv6 extension headers were recognized as “other” protocol. Analysis data of these 2 samples is listed in Table 1, Figure 1 and Figure 2.

2.2 Range of selection

In order to improve the efficiency of random bit-selection, we have to determine the representative range of selection. Since the IPv6 addresses can be separated into prefix and MAC, we can use only one part in our random bit-selection to simplify the process of selection and shorten the simulation time. We designed 5 types of range, as listed below:

Type 1: Prefixes of SA/DA, SP, DP, protocol (168 bits)
Type 2: Full SA/DA, SP, DP, protocol (296 bits)
Type 3: Full SA/DA, SP, DP, but no protocol (288 bits)
Type 4: Prefixes of SA/DA, SP, DP only (160 bits)
Type 5: MAC parts of SA/DA, SP, DP, shortened protocol number (162 bits)

Since there are only 3 major layer-4 protocols in IPv6: ICMPv6, TCP, and UDP. We use 0, 1, and 2 to represent ICMPv6, TCP, and UDP respectively.

Table 1 Analysis of samples 1 and 2

	Sample 1		Sample 2	
	# of packets	# of flows	# of packets	# of flows
ICMPv6	302142	1671	198053	1166
TCP	1589912	6785	1213193	6000
UDP	105447	10344	65345	9841
Other	2499	11	523409	9

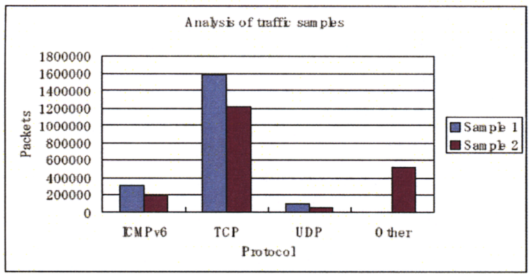

Figure 1 Analysis of traffic samples (in packets)

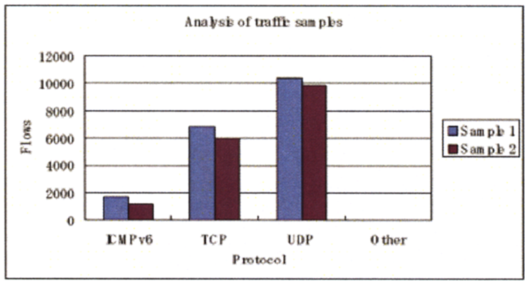

Figure 2 Analysis of traffic samples (in flows)

2.3 Preparation for experiment

We convert the original binary traffic samples to human readable *tcpdump* text format. Further, to collect the flow ID from the samples, we convert the *tcpdump* text format to another text file. These two conversions save storage space and simplify the evaluation processes.

3. Architecture for IPv6 Header Caching

IPv6 has 128-bit IP address with extension headers, which causes packet classification one of the performance bottlenecks. Ternary CAM (TCAM) is the fastest approach of packet classification featuring linear complexity with high cost and slow update. To place 128-bit addresses in TCAM is also very tricky. We propose an approach of caching the recently matched flows. There are three major parameters for cache memory: number of entries, associativity, and replacement policy. The total size of the cache memory is proportional to the number of entries. Associativity decides the number of candidates when replacement occurs. Replacement policy regards the method for choosing a victim to replace. All these parameters affect the performance of cache memory and hardware complexity.

3.1 Cache memory structure

The structure of a cache entry consists of four fields: Index, Tag, TS/Counter, and Result. Each field is described as follows:

Index: It is the entry location in the cache. The index length varies depending on the number of entries and associativity. An index is generated by a variable-length hash function such as random bit-selection. Here the index length is $\log_2$ (No. of entries/Associativity), and the associativity is an integer between 1 and 4.

Tag: It is a 32-bit checksum-like field that checks whether two flows are identical. Tag is generated by a hash function and must be independent of the index to avoid misclassification. Two flows with same index and tag fields are considered as identical.

TS/Counter: It is the recent access record of the entry. Its content is decided by the replacement policy. For LRU (Least Recently Used), this field is used to store a timestamp (TS) as the recent access time; for LFU (Least Frequently Used), it is used to count recent accesses. The length of TS or counter can be determined through the range of timestamp (LRU) or the maximum value of counter (LFU). This field can be ignored if it uses direct-mapping or random replacement policy, or it is in full associativity.

Result: It is the rule number or action for the flow. The length of the result can be derived by either the range of rule number or number of possible actions. With LRU policy, the entry with oldest access time will be replaced. Each cache entry uses TS field, which is a 32-bit packet serial number, to record the last access time. With LFU policy, the entry with least access count will be replaced. Each cache entry includes a counter which increments with a cache hit, and reset to zero with a cache miss. With random replacement policy, the victim to be replaced is randomly chosen through the generated random number which is an integer between 0 and associativity N-1.

3.3 Associativity

The number of candidates to be replaced depends on the associativity, which is an important parameter for cache memory because it affects the performance and complexity. With full associativity, all entries in the cache memory are candidates. The performance of full associativity is already known as the best, but the hardware cost is very high because a large number of logic gates are required to do comparison in parallel. In full associativity, index field is ignored when matching a flow or replacing an entry. With N-way associativity, the number of candidates to be replaced is equal to N. It is called direct-mapping if N equals to 1, and only one candidate needs to be replaced, thus the replacement policy does not affect the result in direct-mapping.

3.4 Hash functions

We proposed three hash functions to generate the field "Tag" in cache memory and tested them with the aforementioned two traffic samples. Figure 3 shows hash function I, in which the protocol field is ignored. If the layer-4 protocol is neither TCP nor UDP, we set both source port and destination port number to zero. The hash function can be implemented directly with XOR gates. It contains 3 functions: reverse, combine and split. The total delay of hash function I is three times XOR gate delay. The hash function II is very similar to hash function I, but the port number was set to 65,535 if layer-4 protocol is neither TCP nor UDP. The figure of hash function II is omitted here. Figure 4 shows hash function III, which is similar to both I and II with XOR gates replaced by XNOR gates. Same as II, port number is set to 65,535 if layer-4 protocol is neither TCP nor UDP. The total delay of hash function III is three times XNOR gate delay. We evaluated three hash functions with two traffic samples mentioned before. The collision rates should be as low as possible to avoid misclassification. Evaluation results are listed in Table 2, in which collision rates of hash function II and III are much higher than function I for sample 1. But collision rates of the three functions for sample 2 are quite similar. So we suggest using hash function I to achieve lower collision rate for both samples.

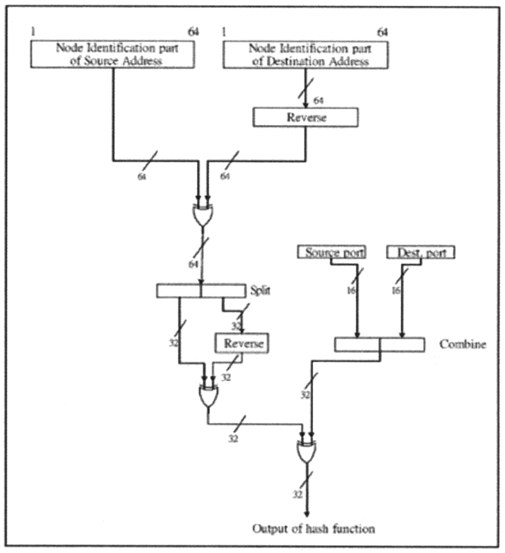

Figure 3 Hash function 1

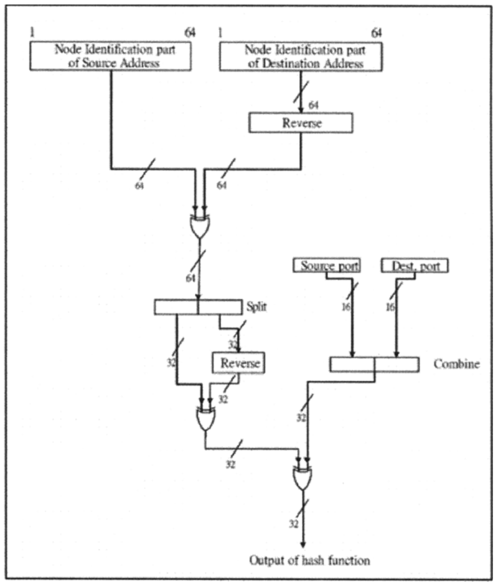

Figure 4 Hash function 3

Table 2 Collision analysis of three hash functions

Sample / Hash function	Sample 1		Sample 2	
	No. Flows	Collision rate	No. Flows	Collision rate
Hash function I	2425	12.8%	1884	10.5%
Hash function II	4690	24.9%	1549	9.1%
Hash function III	4797	25.5%	1844	10.8%

3.5 Rule updates

Issues of rule update are rarely discussed in packet classification speedup techniques. We recommend storing rule number instead of rule action in the result field. For updates to the prefix or port number of a rule numbered X, its associated cache entries should be cleared to avoid inconsistency. For updates to the action of a rule numbered X, its associated cache entries should not be changed.

4. Experimental Measurements

4.1 Simulation of random bit-selection

Our experimental procedures are as follows. (1) Translate the original binary traffic sample to text format with *tcpdump*. (2) Extract the essential fields from the text format sample to another file. (3) Compute collision counts, number of addresses, and number of flows from the extracted file.

In order to optimize the performance of random bit-selection, the ranges of selection should be chosen carefully. The types of range are listed in Section 2.2. For sample 1, the collision rates are between 55% and 29%, as shown in Table 3, and the number of collisions is shown in Figure 5. For 24 or 32-bit length, type 2 is the best choice. For 8 or 16-bit length, type 5 is the best choice. There is only little difference when the length of random bit-selection is 32. For sample 2, the collision rates are between 17% and 6%, as shown in Table 4, and the number of collisions is depicted in Figure 6. For 24 or 32-bit length, type 2 is the best choice. For 8 or 16-bit length, type 5 is the best choice.

Simulation results show that differences between the types are relatively small. We can use any type if the difference is negligible. Smaller ranges such as type 1, 3, 4, and 5, can simplify the hardware design and shorten the simulation time.

Table 3 Random bit-selection on sample 1

Type \ Bits	8	16	24	32
Type 1	43.1%	39.8%	30.0%	32.1%
Type 2	52.7%	37.7%	30.4%	28.7%
Type 3	50.5%	36.0%	33.8%	32.7%
Type 4	46.1%	49.3%	29.0%	31.7%
Type 5	44.7%	35.4%	33.1%	30.5%

Table 4 Random bit-selection on sample 2

Type \ Bits	8	16	24	32
Type 1	16.7%	11.0%	9.6%	8.8%
Type 2	14.5%	10.9%	7.4%	5.7%
Type 3	14.4%	10.2%	6.9%	7.1%
Type 4	13.7%	11.5%	9.2%	9.4%
Type 5	14.1%	10.7%	9.2%	7.2%

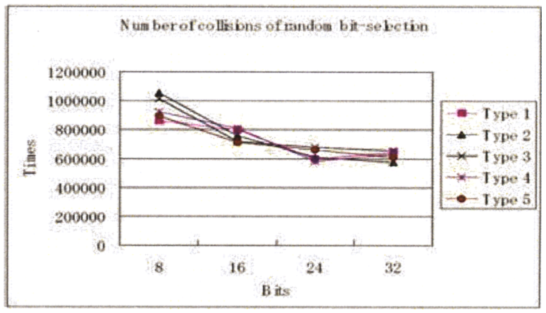

Figure 5 Random bit-selection on sample 1

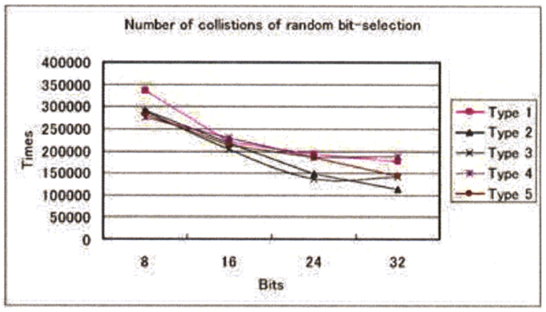

Figure 6 Random bit-selection on sample 2

4.2 Simulation of IPv6 header caching

We use sample 2 for our experiment of IPv6 header caching. To simplify the experiment, packets belonging to "other" category were omitted. So the total number of packets is 1,476,591, instead of 2,000,000. We use random bit-selection with type 1 range, and hash function I in our simulations. We use type 1 range to complement the fields that hash function I ignores.

We evaluate the miss rate with different replacement policies, associativities, and cache sizes. Simulations on IPv6 header caching take much longer time than random bit-selection. Simulations of full associative cache memory take the longest time because the indices generated by random bit-selection are ignored, and almost all lookups are performed with linear search. Experimental results are listed in Table 5 and 6.

From the results, the miss rate is low even if the number of cache entries is 256 and associativity is one. Increasing associativity or cache size causes miss rate to decrease. Replacement policy does not affect the miss rate in direct-mapping, so the miss rates of different replacement policies are the same. Evaluation results of full associativity with different replacement policies in Table 5 are omitted, because the number of total flows is below 18,000, which is much less than 65,536. Miss rate differences among the replacement policies are very small, so LRU is set to the default replacement policy in our latter evaluations. Evaluation results of full associativity in Table 6 are below 10%, most results are below 5%. Miss rates with associativity 4 and more than cache 1024 entries are also below 10%. This means we can speedup the process of packet classification with a small size of cache memory.

Misclassification means distinct flows are judged as the same one. Since hash function I is independent of random bit-selection, misclassification can be avoided completely because we compress the flow ID to 32 bit or less. To evaluate misclassification ratios, we place an extra field "flowID" in each cache entry to record original uncompressed flow ID and check whether any two flows with same index and tag values are identical. Evaluation results show that increment of associativity increases misclassification rate because the length of index decreases with the increase of associativity. So associativity should be chosen carefully when cache size is fixed. There is a trade-off between misclassification ratios and miss ratios. Fortunately, misclassification ratios are all below 5% in our experiment. As shown in Table 7, in direct-mapping, misclassification ratio is only 1.7%. So the probability of misclassification in practical packet processing is very little. Regarding the hardware complexity, each cache entry contains a 32-bit Tag, an N-bit TS/Counter, and an M-bit Result. There are (32+N+M)*X bits in a cache memory with X entries. Hardware cost increases with N and M.

Table 5 Miss rates under different replacement policies and associativities with 65536 entries

Policy / Assoc.	LRU		LFU (limit=4)		Rand	
	Times	Rate	Times	Rate	Times	Rate
1	218643	14.8%	218643	14.8%	218643	14.8%
2	131025	8.8%	158898	10.8%	164723	11.2%
4	126149	8.5%	116147	7.9%	130646	8.8%

Table 6 Miss rates with different associativities and cache sizes under LRU replacement policy

Entries / Assoc.	256		1024		4096		16384	
	Times	Rate	Times	Rate	Times	Rate	Times	Rate
Full	109798	7.4%	33500	2.3%	18957	1.3%	15155	1.0%
1	296053	20.0%	232061	15.7%	233390	15.8%	204623	13.9%
2	260528	17.6%	220875	15.0%	186072	12.6%	165337	11.2%
4	201731	13.7%	138976	9.4%	129156	8.7%	156062	10.6%

Table 7 Misclassification ratios with different replacement policies and associativities

Policy / Assoc.	LRU		LFU		Rand	
	Times	Rate	Times	Rate	Times	Rate
1	25082	1.7%	25082	1.7%	25082	1.7%
2	58813	4.0%	65179	4.4%	54426	3.7%
4	71486	4.8%	74138	5.0%	71673	4.9%

5. Conclusions and Future Works

We proposed a cache architecture and hash functions to improve the performance of packet classification for IPv6. Cache miss ratios in our evaluations are pretty low even if cache size is small. Misclassification ratios are also low enough. High-speed packet processing in future IPv6 environment can be achieved with a cache memory at reasonable cost. In IPv6, options of IP are stored in extension headers. There are some packets with IPv6 option headers in our samples, but *tcpdump* cannot parse all the extensions with default buffer setting. To find the header of the layer-4 protocol for packet classification, routers must spend more time to trace the protocol chain. If the process of packet classification uses some speed-up mechanisms such as pipeline, the extension headers will stall the pipeline and affect the throughput. Routers must also process the extension headers. It will be a big performance problem in practical IPv6 packet processing. There are some special layer-4 protocols other than ICMPv6, TCP, and UDP in our samples. They are IP-in-IP (tunneling), PIM, and some other special layer-4 protocols in real Internet traffic. For IP-in-IP, there is another IP header encapsulated in the packet, and the header of layer-4 protocol is behind the encapsulated IP header. We have to decide whether to process the encapsulated IP header or not. Further, we have to set the default action for all protocols not mentioned before. Flow label is a 24-bit field in IPv6 header to identify a flow effectively. We may use it to speed up the packet classification process if it is widely used in the future.

References

[1] P. C. Wang et al., "Performance improvement of packet classification by using look-ahead caching," IEICE Trans. on Comm., vol. E87-B, no. 2, pp. 377-379, Feb. 2004.

[2] P. Gupta and N. McKeown, "Algorithms for packet classification," IEEE Network, vol.15, no.2, pp.24-32, Mar/Apr 2001.

[3] S. Iyer, R. R. Kompella, and A. Shelat, "ClassiPI: An architecture for fast and flexible packet classification," IEEE Network vol.15, no.2, pp.33-41, Mar/Apr 2001.

[4] M. Uga and K. Shiomoto, "A novel ultra high-speed multi-layer table lookup method using TCAM for differentiated services in the Internet," IEEE Workshop on High Performance Switching and Routing, pp. 240-244, May 2001.

[5] K. Li, et al., "Architectures for packet classification caching," The 11th IEEE International Conference on Networks, pp. 111-117, Sept. 28-Oct. 1, 2003

[6] P. Gupta and N. McKeown, "Classifying packets with hierarchical intelligent cuttings," IEEE Micro, vol. 20, iss. 1, pp.34-41, Jan/Feb 2000.

[7] C. Williamson, "Internet traffic measurement," Internet Computing, vol. 5, no. 6, pp. 70-74, Nov/Dec 2001

[8] http://tracer.csl.sony.co.jp/mawi/